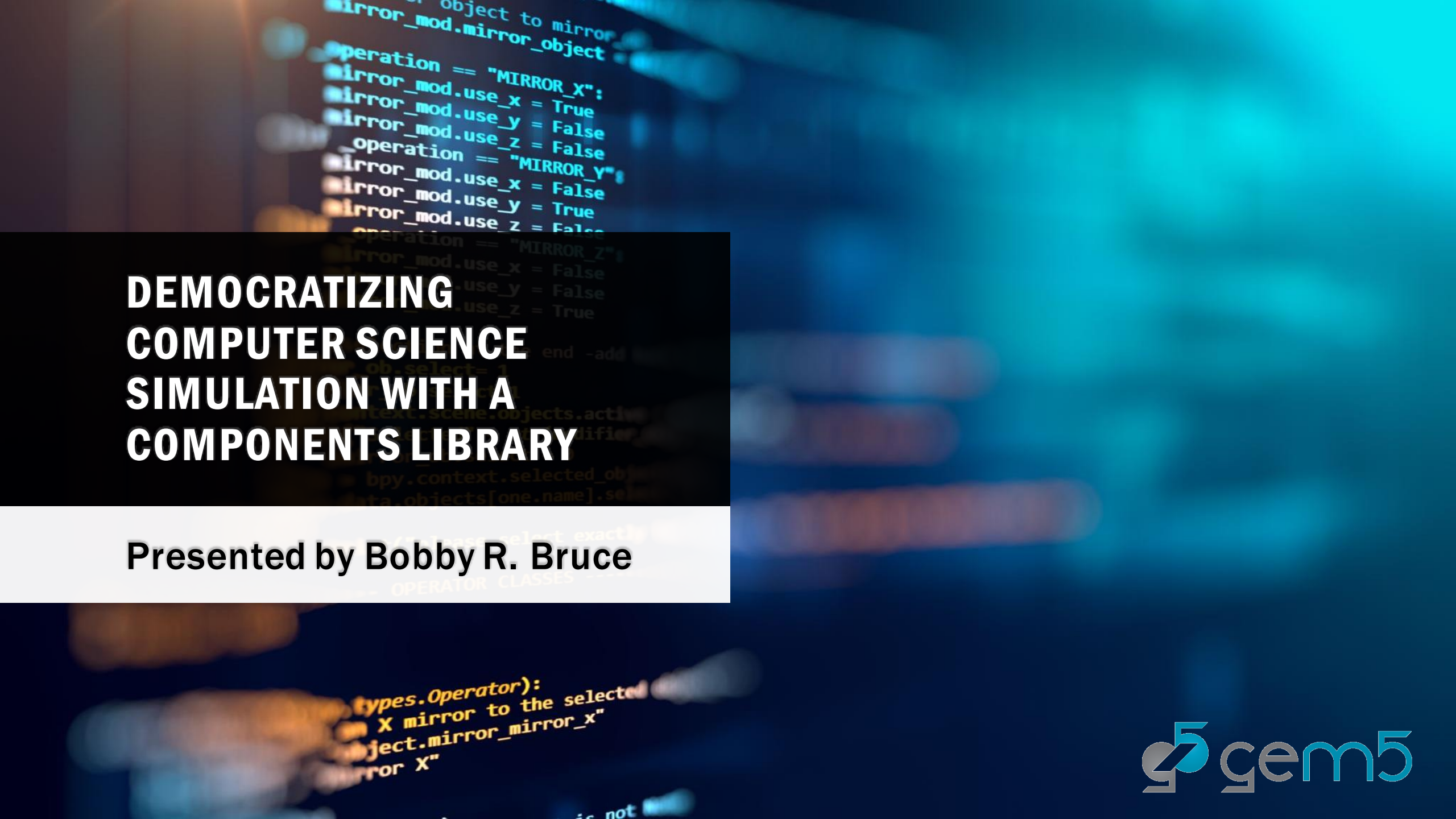


DEMOCRATIZING COMPUTER SCIENCE SIMULATION WITH A COMPONENTS LIBRARY

Presented by Bobby R. Bruce

gem5

"HELLO WORLD" IN GEM5: PLENTY OF BOILERPLATE

simple.py

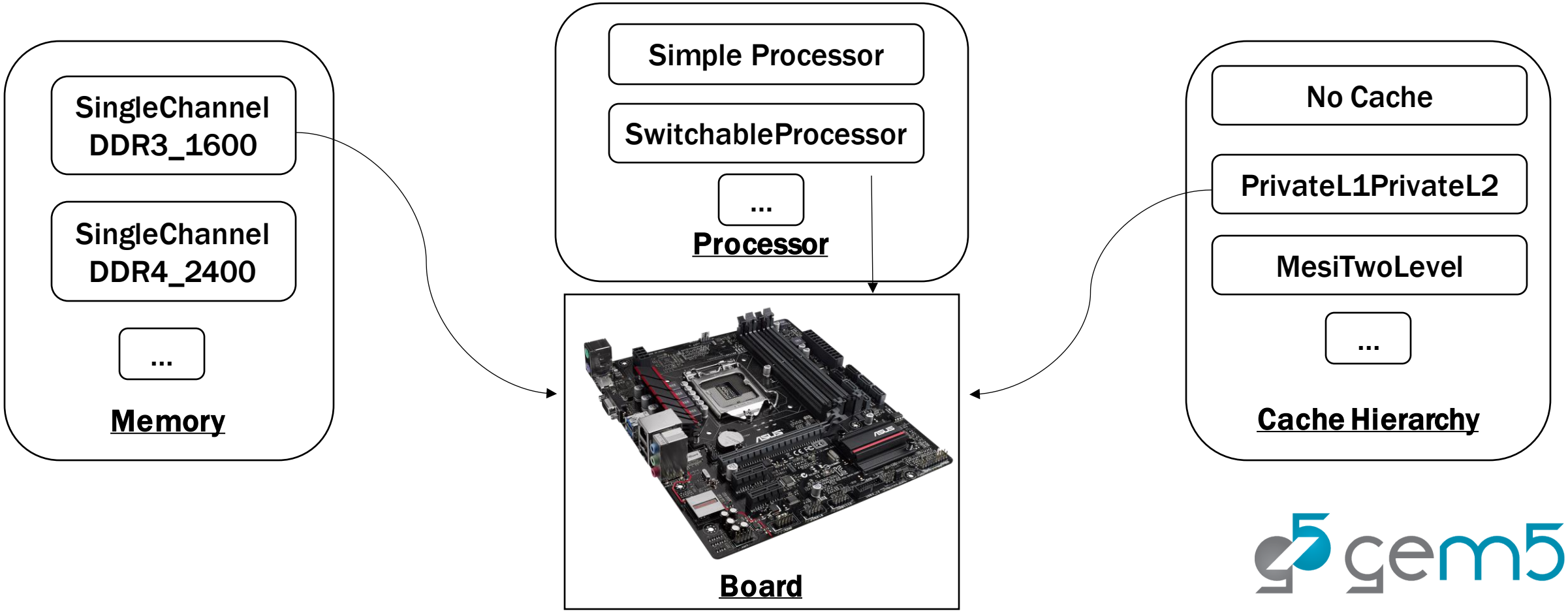
simple.py > No Selection

```python
import m5
# import all of the SimObjects
from m5.objects import *

# create the system we are going to simulate
system = System()

# Set the clock fequency of the system (and all of its children)
system.clk_domain = SrcClockDomain()
system.clk_domain.clock = '1GHz'
system.clk_domain.voltage_domain = VoltageDomain()

# Set up the system
system.mem_mode = 'timing'               # Use timing accesses
system.mem_ranges = [AddrRange('512MB')] # Create an address range

# Create a simple CPU
system.cpu = TimingSimpleCPU()

# Create a memory bus, a system crossbar, in this case
system.membus = SystemXBar()

# Hook the CPU ports up to the membus
system.cpu.icache_port = system.membus.cpu_side_ports
system.cpu.dcache_port = system.membus.cpu_side_ports

# create the interrupt controller for the CPU and connect to the membus
system.cpu.createInterruptController()

# For x86 only, make sure the interrupts are connected to the memory
# Note: these are directly connected to the memory bus and are not cached
if m5.defines.buildEnv['TARGET_ISA'] == "x86":
    system.cpu.interrupts[0].pio = system.membus.mem_side_ports
    system.cpu.interrupts[0].int_requestor = system.membus.cpu_side_ports
    system.cpu.interrupts[0].int_responder = system.membus.mem_side_ports

# Create a DDR3 memory controller and connect it to the membus
```

- A single core setup connected directly to main memory, with no cache, requires 36 lines of Python!
- Many hundreds are required for a system capable of booting a modern OS.
- Unsupported scripts and examples are circulated in the community as many configurations do not vary between simulations.

gem5

SOLUTION: GEM5 COMPONENTS



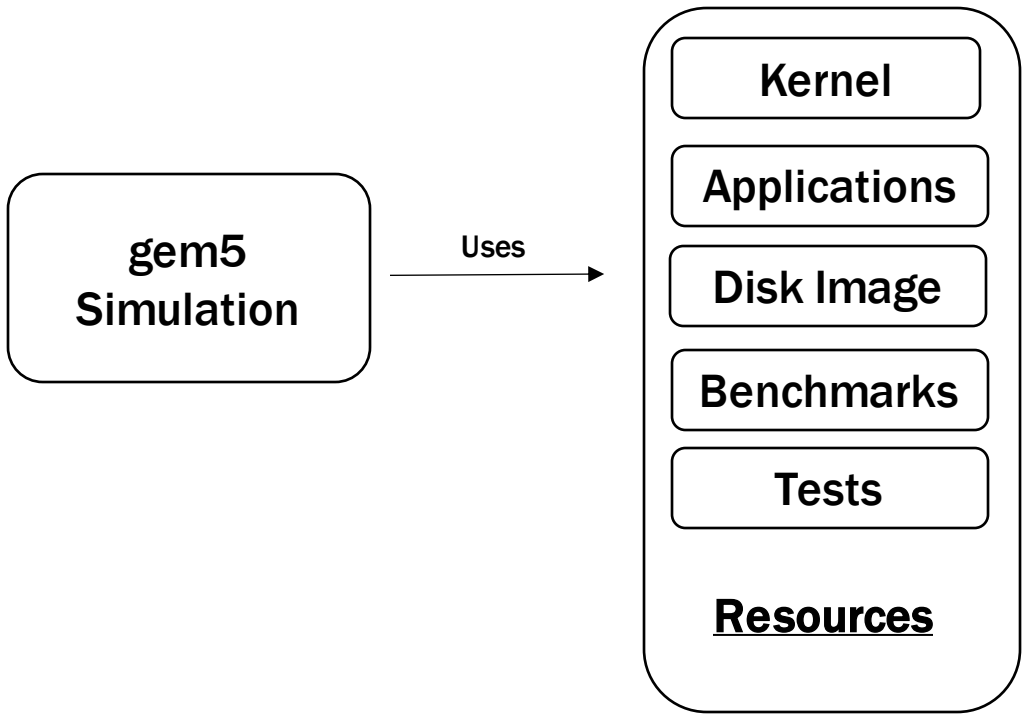
GEM5 COMPONENTS

gem5components.py

gem5components.py 〉 No Selection

```python
from gem5.components.boards.x86_board import X86Board
from gem5.components.memory.single_channel import SingleChannelDDR3_1600
from gem5.components.processors.simple_processor import SimpleProcessor
from gem5.components.cachehierarchies.classic.private_l1_cache_hierarchy import (
    PrivateL1CacheHierarchy,
)
from gem5.processors.cpu_types import CPUTypes

memory = SingleChannelDDR3_1600(size="3GB")
cache_hierarchy = PrivateL1CacheHierarchy(l1d_size="16kB", l1i_size="16kB")
processor = SimpleProcessor(cpu_type=CPUTypes.TIMING, num_cores=4)

board = X86Board(
    clk_freq="3GHz",
    processor=processor,
    memory=memory,
    cache_hierarchy=cache_hierarchy,
)

board.connect_things()
```

AND, GEM5 RESOURCES!



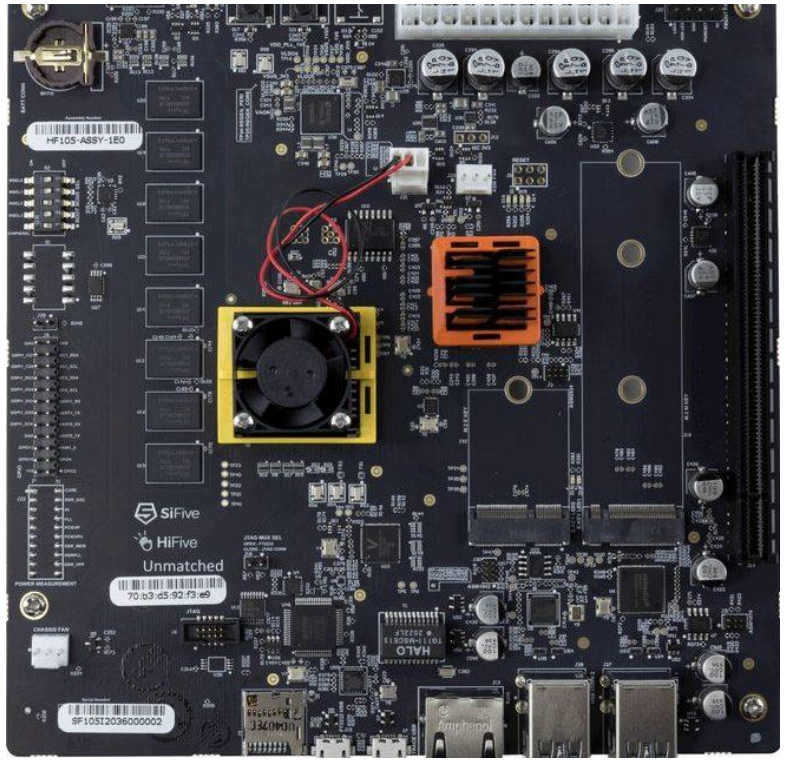
Uses

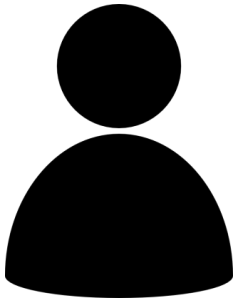
resources.py

```python
from gem5.components.board.x86_board import X86Board
from gem5.resource import Resource
from gem5.simulate import simulate

board = X86Board(...)

board.set_workload(
    kernel = Resource("x86-linux-kernel-5.4.49"),
    disk_image = Resource("x86-ubuntu-img"),
)

simulate(board)
```

- gem5 resources contains pre-built resources for gem5 simulations.
- They may be automatically obtained via configuration scripts.

COMING SOON! "KNOWN GOOD CONFIGURATIONS"





Known Good Configurations

sifive_board.py

```
1  from gem5.known_good.unmatched_sifive import SiFive
2  from gem5.simulate import simulate
3  from gem5.resource import Resource
4
5  board = SiFive()
6
7  board.set_workload(
8      Resource("riscv-ubuntu-boot")
9  )
10
11 simulate(board)
12
```